PAS 2.0 Communication Architecture

William .D. Ivancic[a,*], Oben S. Sands[a], Casey J. Bakula[a], Ted Wright[a], Martin A. Bradish[a]

[a]*NASA Glenn Research Center, 21000 Brookpark Road, Cleveland, Ohio 44135 USA*

Abstract

This this document describes the communication architecture for the Power, Avionics, and Software (PAS 2.0) subsystem for the Advanced Extravehicular Mobile Unit (AEMU). the following systems are described in detail: Caution Warning and Control, Informatics, Storage, Video , Audio, Communication, and Monitoring Test and Validation. this document also provides some background as well as the purpose and goals of the PAS project at NASA Glenn Research Center.

Keywords: Communication Architecture, Protocols, Telemetry, Avionics

Contents

*Corresponding author

Email addresses: william.d.ivancic@nasa.gov (Willian .D. Ivancic), oben.s.sands@nasa.gov (Oben S. Sands), casey.j.bakula@nasa.gov (Casey J. Bakula), ted.wright@nasa.gov (Ted Wright), martin.a.bradish@nasa.gov (Martin A. Bradish)

1. Background

NASA is currently developing an Advanced Extravehicular Mobility Unit (AEMU)[1] under the Advanced Exploration Systems Program. A key part of this development is the spacesuit Primary Life Support Subsystem (PLSS) technology unit that is human-rated for long-duration microgravity or planetary missions, and vacuum or low-pressure environments.

The Current EMU's (space suits) are still using technology from the 1970's and 1980's particularly regarding communications system. These systems work well and there is always a reluctance to change for space systems and nearly always the desire to be backwards compatible with proven systems, reduce risk, and save cost. Unfortunately, this backwards compatibility often occurs to the detriment of infusing new technologies. Ironically, infusing new technologies may actually reduce cost while dramatically improving capability.

The Power, Avionics, and Software (PAS) project at Glenn Research Center (GRC) support the Advanced Extravehicular Mobile Unit (AEMU) program at NASA's Johnson Space Center (JSC). GRC's role is developed a prototype Command Warning and Control (CWC) subsystem and to research new technologies for the AEMU. The results will be used to develop the necessary requirements for the AEMU as well as to potentially integrate new technologies into the AEMU. GRC personnel are also working with potential suit contractors to provide lessons learned from these technology developments and research.

2. Documentation

In order to reduce cost, in increase productivity, the PAS Communication Architecture group utilized modern documentation and collaboration tools. Within much of NASA, the current widely practiced method of handling issues, requirements and general documentation is to utilize spreadsheets and pass various versions around with one person in control. A "most-up-to-date" version is often kept on a collaborative server such as EMC Corporation's eRooms or Microsoft's SharePoint®. The problem is the documents on those servers are rarely the current working documents. Rather, there are three or four or five or more working documents and when changes are made, nobody knows who made them, when or why. Thus the PAS Communication Architecture group adopted techniques used by software programmers including use of Issue Trackers and Wiki. This ensure all documents are in one place and up-to-date with most of the material directly on the wiki and any critical issues documented in the issue tracker. Use of revision control (version control) has been implemented for all documentation, not just software. Thus, users know what the latest documents are as well as what revisions have been made, when and by who. The same can be said regarding Issues and general documentation residing on the wiki.

[1] A spacesuit that provides environmental protection, mobility, life support, and communications for astronauts performing an extra-vehicular activity (EVA)

3. Terminology

To aid in discussion within this document it is useful to develop and define some terminology specific to our concepts of operations.

Caution and Warning Events

Our preferred caution and warning definitions are below. We prefer these to past definitions a they align with industry practices, such as terminology used in industrial plants and power plants. These definitions were altered in the Intersystem Communications Section in order to maintain consistency with terminology currently in use on the International Space Station and previously used on the Shuttle (Space Transportation System - STS).

Alert Single Tone for minor events such as "Text Message Received" or when a suits operational configuration has changed.

Caution Take notice, but don't take action (usually flashing lights in the power plant).

Warning Take Action Now. (Audio Alarms. Takes precedence over any cautions).

Data Types There are four basic data types listed below. Note, there are not command types or response types. Rather, we use events and telemetry. The interaction between subsystems is based entirely on a publish/subscribe concept that greatly simplifies code and reduces the amount of state that one needs to maintain.

Application (e.g., text messaging, file transfer)

Event - a message type indicating change of state that my require some action. Change of state includes items such as caution and warning signals, audio push buttons positions, and call group selections.

Streaming - off suit real-time streaming (e.g, audio and video)

Telemetry (e.g., system and subsystem status, configuration and measurements)

Home Home is where the crew members return to after their EVA (e.g. ISS, Habitat, Base Camp, Tier-1 Relay in the Network Diagram)

Telemetry Often telemetry refers to ALL data coming from a spacecraft to the ground. For the AEMU, telemetry is just system and subsystem status, configuration and performance measurement data (e.g. suit internal pressure, suit internal temperature, power supply voltage, power supply current, radio transmission rate). All other data flows such as file transfers, and streaming video and audio are considered data flow and not telemetry.

Call Group (i.e. Voice Loop) A group of voice reception devices that all listen on a common "channel". For Voice-over-IP (VOIP) that channel may be a mutlicast address and port. For simple RF radio communications that channel may a particular frequency and time slot or spreading code. For illustration purposes assume we have 3 AEMUs. Then, some examples of call groups for AEMU may be:

- All three AEMUs,
- A single AEMU to Home and Mission Control,
- All three AEMUs, Home and Mission Control, and,
- Home and Mission Control.

4. Communication Architecture Design Philosophy

The goal is to to develop a Flexible, Extensible, Testable, and Cost Effective Architecture. The design is not just for use on the International Space Station (ISS) or Orion, but for future planetary exploration science missions' terrestrial sorties. There is also a great desire to be able to readily infuse new technology.

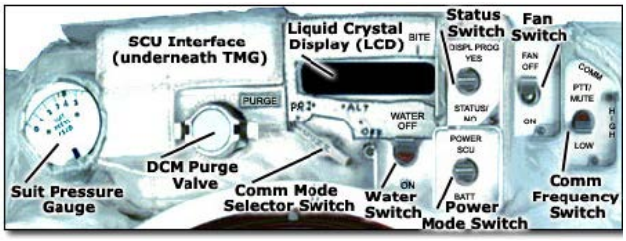

Figure 1: Power, Avionics and Control Software Critical Systems

Subsystems have developed, for the most part, independently. This was done for a number of reasons. First and foremost, CWC is a critical system that is being develop with direct path to flight in mind. Thus, there is a desire to keep both the hardware and software is simple and reliable as possible while maintaining power and mass requirements. The audio processing and communication subsystems both have significant research elements. Allowing these systems to be developed independently enables the research elements to not have to rely on other subsystems and visa versa. The informatics system is highly desirable for AEMU sorties involving scientific research and terrestrial exploration. As such, it is considered a long-term development.

Currently, each of the subsystems has its own processing unit. A future AEMU would likely have only two or three processors. The CWC would likely maintain its own processor while the rest of the systems might share one. The communication. and audio processing system may be combined having a separate processing unit. Informatics may have its own high-end processor and software as this unit is likely to be deployed last and updated often. This should not be a problem as Informatics is a "non-critical subsystem" [2].

NASA's current mission and budget profile projects the AEMU development to occur in a staged, evolutionary process. One could anticipate that the CWC, the audio, and the communications subsystems would be developed prior to informatics. This is particularly true since informatics provides its greatest return on investment for exploration sorties rather than EVA maneuvers around International Space Station or the Orion spacecraft.

Under PAS-2.0 there is neither sufficient funds, people, or time to reinvent what already exists. Thus, new technologies and techniques have been adopted such as Google Protocol Buffers for telemetry, ZeroMQ® for messaging, and Wireshark® for monitoring activity across the GigE and WAN busses.

There is also a great desire to learn from others experiences. For example: the AEMU sortie operations are very similar to soldier operations as are many of the requirement and features. Both the army operations and the AEMU need separate call groups and multi-homed radio systems. The Soldier Radio Waveform already has these capabilities. The AEMU is also quite similar to Intelligent transportation system (ITS) vehicles with some system being critical for safety of life and others being desired features. In fact, in many ways, ITS has more stringent requirements as security is a major concern.

5. Switches and Controls

On the current EVA suits, the Display and Control Module (DCM) is where the EVA crewmember interacts with the Primary Life Support Subsystem (PLSS). It contains displays and controls for the operation of the EMU and is attached to the front of the upper torso of the EMU. In order to read the displays at this location, the crewmember uses mirrors (attached to the arms of the EMU) to reflect and read the information, which is written backward on the front of the suit. Sensor readings are provided to the crew and read on the top of the DCM on the Liquid Crystal Display (LCD) [1] - see figure 1.

For the prototype AEMU, external controls such as switches and knobs are associated with individual subsystems. For example, the push to talk switch (PTT) and the volume control switch are associated switch with the audio subsystem. The call group knob (rotary switch), similar to that shown for Temperature control in figure 2, is associated

[2]Non-Critical here means that if the system fails, the sortie may have to be terminated of may be less than fully successful, but that there is no danger of loosing the AEMU.

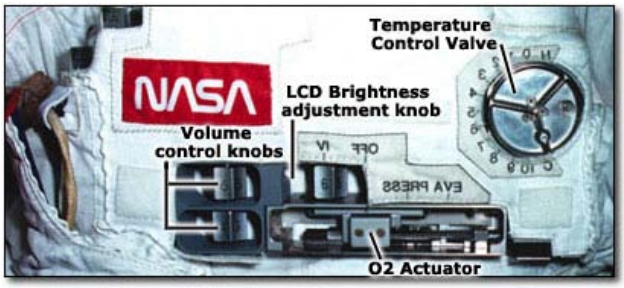

Figure 2: Power, Avionics and Control Software Critical Systems

with the communication subsystem. In this instance, are no commands either internal from other subsystems or remote to set the volume. It is performed by the audio subsystem via the volume switch. However, there is a desire to to display the volume settings. As such, a telemetry message indicating that information is periodically sent from the audio subsystem to whatever subsystems or remote systems desire that information. In a similar manner, the communication system creates a telemetry message indicating the current call group setting (knob position).

6. Telemetry

Our desire is to develop a flexible telemetry format that does not required fixed fields predefined full telemetry formats to be sent each transmission. Rather, we desire to send only that telemetry that the subsystem ascertains is necessary or that the authorized requesting entity wishes to obtain.

We investigate fix ID/Data pairs such that each piece of telemetry is identified by a Flag indicating if the next field is value is an ID or data. The ID value would indicate what the data represents (e.g. voltage, pressure, Bit-error-rate, etc...). The data value would represent the value corresponding to the ID (e.g., Boolean, Integer, etc...). This format meet our desires. However, upon further investigation, we decided to utilize Google Protocol Buffers (GPB or simply "protocol buffers") [2]. The telemetry format is shown in figure 3. We can run protocol buffers over UDP if so desired. Currently are implemented all messaging using ZeroMq (TCP transport mode) for ease of development in a prototype system - see section Intersystem Communication. Protocol buffers uses a key/value pair for each entry. These key/value pairs utilize Varints, a method of serializing integers using one or more bytes. This method allows for very efficient bits-on-the-wire encoding. Furthermore, protocol buffers allows for optional key value pairs. In our deployment, the only required key/value pairs are timestamp, sender (e.g, CWC, INFO, AUIDO, COMM) and type. All other key/value pairs are optional enabling great flexibility.

6.1. Bits on the Wire

In addition to the protocol buffer payload, the packet wire format contains zero to three null (0x00) padding bytes appended to the end to make the total ZeroMQ data size a multiple of 32 bits. This is needed to preserve compatibility with older ARM microprocessors that do not have native instructions for byte alignment. The protocol buffer payload is prefixed by a 16 bit field that stores the length of the payload in bytes. A 16 bit Cyclic Redundancy Check (computed over the combined length, payload and pad bytes) is added before the length field. The CRC and length are stored in network byte order. Thus we have:

2 byte CRC calculated over of the rest of the packet (network bye order) 2 bytes which is the length of the encoded protocol buffer (network bye order) A variable length encoded protocol buffer payload 0 to 3 pad bytes (0x00) to make the total packet length a multiple of 4

6.2. Google Protocol Buffers

Use of GPB greatly simplifies coding as all header information and associated data structures and code is automatically generated from the GPB ".proto" file. The following excerpt is taken directly from the GPB web pages:

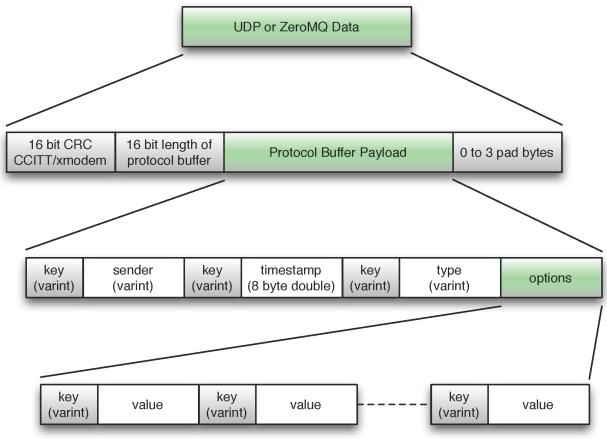

Figure 3: Protocol Buffer Telemetry Format

Protocol buffers were initially developed at Google to deal with an index server request/response protocol. Prior to protocol buffers, there was a format for requests and responses that used hand marshalling/unmarshalling of requests and responses, and that supported a number of versions of the protocol. This resulted in some very ugly code. Explicitly formatted protocols also complicated the rollout of new protocol versions, because developers had to make sure that all servers between the originator of the request and the actual server handling the request understood the new protocol before they could flip a switch to start using the new protocol.

Protocol buffers were designed to solve many of these problems:

- New fields could be easily introduced, and intermediate servers that didn't need to inspect the data could simply parse it and pass through the data without needing to know about all the fields.
- Formats were more self-describing, and could be dealt with from a variety of languages (C++, Java, etc.)

As the system evolved, it acquired a number of other features and uses:

- Automatically-generated serialization and deserialization code avoided the need for hand parsing.
- In addition to being used for short-lived RPC (Remote Procedure Call) requests, people started to use protocol buffers as a handy self-describing format for storing data persistently (for example, in Bigtable).
- Server RPC interfaces started to be declared as part of protocol files, with the protocol compiler generating stub classes that users could override with actual implementations of the server's interface.

Protocol buffers are now Google's lingua franca for data at time of writing, there are 48,162 different message types defined in the Google code tree across 12,183 .proto files. They're used both in RPC systems and for persistent storage of data in a variety of storage systems.

Since GBP if so widely used, additional tools have been created by third party developers.

- nanobp is a plain-C implementation of Google's Protocol Buffers data format. It is targeted at 32 bit microcontrollers, but is also fit for other embedded systems with tight (2-10 kB ROM, ¡1 kB RAM) memory constraints.[3]
- protobuf-wireshark is an auto-generate Wireshark dissector plugins for Protocol Buffer messages.[4][2]

6.3. Messages

Since the AEMU PAS-2.0 is a prototype with the results used to develop the necessary requirements for the AEMU as well available for consideration in the development of the flight implementation, we concentrated on developing

the required messages and human readable code and not overly concerned with the efficiency of bits-on-the-wire - particularly over a GigE bus. Our current strategy to accomplish this is to utilize ZeroMQ [5]. ZeroMQ (a.k.a., ZMQ or 0MQ) "is a messaging library, which allows you to design a complex communication system without much effort." - Nicholas Pil. This is one major reasons for our use of ZMQ. We feel it is far more valuable in this prototype effort to concentrate on the Messages (what needs to be communicated between subsystems and AEMUs) rather than the Messaging (implementation). Furthermore, it may result in the future AEMU actually deploying this code rather than writing their own. The following summary of ZMQ features some of which are taken from the ZMQ web site and user guide [5].

- Moves data between networked subsystems regardless of its format.
- Faster than TCP, for clustered products and supercomputing.
- Has excellent support for the publish/subscribe network architecture.
- Carries messages across inproc, IPC, TCP, and multicast.
- Connect N-to-N via fanout, pubsub, pipeline, request-reply.
- Asynch I/O for scalable multicore message-passing apps.
- Large and active open source community.
- 30+ languages including C, C++, Java, .NET, Python.
- Most OSes including Linux, Windows, OS X.
- Free, EASY, and widely used (i.e. Storm, dotCloud, Mongrel2, ZeroRPC, SaltStack) Easy means we can concentrate on the data being sent rather than implementation details.
- GNU [3]Lesser General Public License (LGPL) free software [6]

A Publish/Subscribe network architecture is a best practice for reliable information exchange between loosely coupled networked subassemblies, such as the AEMU. While it is possible to create a simple system based on UDP broadcasts, past experience has shown that making it reliable and fixing all the corner cases can become complicated. Building distributed software and getting code talking to code can be difficult and expensive. ZeroMQ makes this task cheap and easy or at least easier. The open source ZeroMQ socket library makes coding a basic Publish/Subscribe system trivially easy, requiring about a dozen lines of code. ZeroMQ is a library that is linked to code, so unlike most other "middleware" solutions, it does not require a separate daemon process or broker software. It was originally developed for high frequency trading applications on Wall Street, where efficiency is paramount.

6.4. Software

In the PAS-2.0 effort, there is a great desire to develop subsystem software for reuse. Using Google Protocol Buffers and ZeroMQ enable this.

With GPB one can easily add key/value pairs without effecting existing code.

Using ZeroMQ one can easily switch a transport mechanism as it uses different protocols depending on the peers location (TCP, PGM multicast, IPC, inproc shared memory) [7] which is extremely importantly for our code reuse. One can easily switch from and IP routing to interprocess control (IPC) or in process shared memory. Thus, if the bus structure changes from GigE to Peripheral Component Interconnect (PCI) bus or some other bus, one simply needs to change the transport type in ZeroMQ code and not much else (e.g., in the italicized line below, tcp become pgm for multicast and ipc for interprocess control).

[3]GNU is a recursive acronym for GNU's Not Unix!. The Hurd, GNU's own kernel, is some way from being ready for daily use. Thus, GNU is typically used today with a kernel called Linux. This combination is the GNU/Linux operating system. GNU/Linux is used by millions, though many call it "Linux" by mistake.

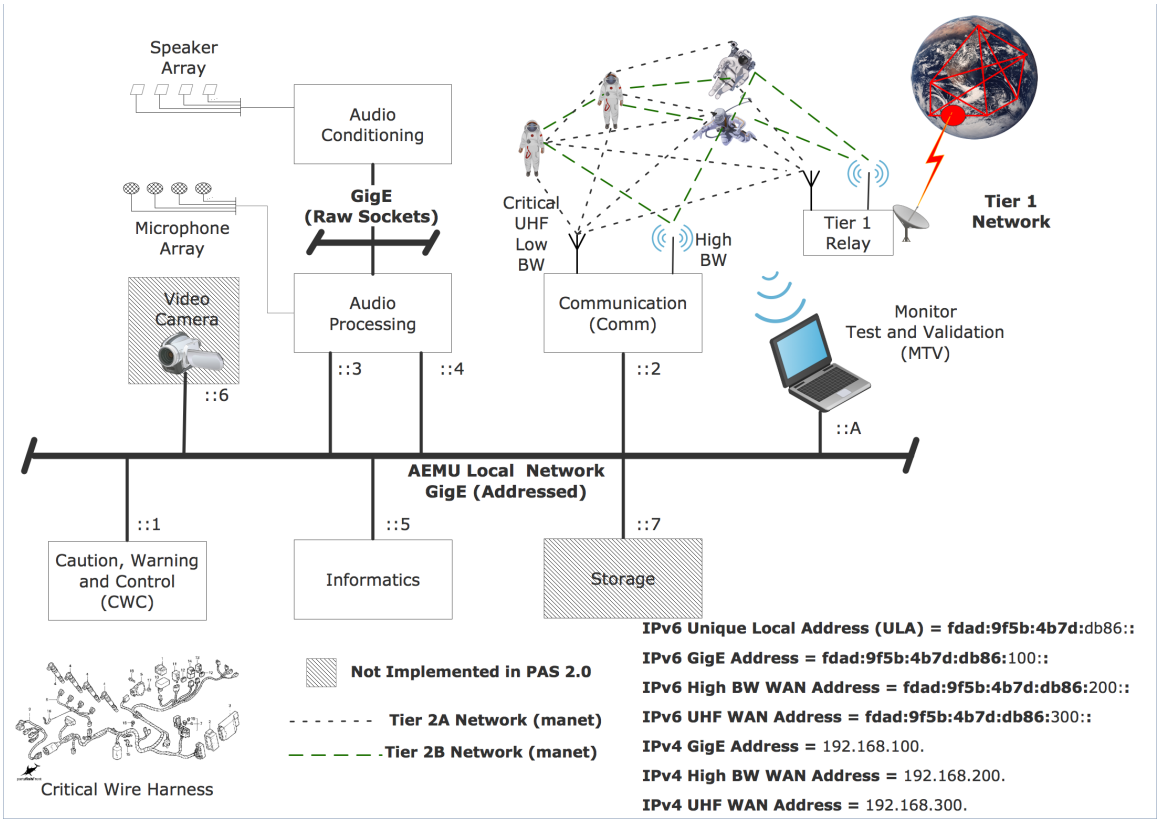

Figure 4: Power, Avionics and Control Communication Architecture

*subSockets.connect(**tcp**://localhost:%d' %pubPort)*

Software was developed in a manner that allows each subsystem to draw upon a common library of messages This enables the subsystem software to remain intact even if the bus architecture changes.

Originally we were going to implement a common set of APIs (Application Program Interfaces). However using ZMQ and GPB, the messaging was so simple, that there was no need for actual API development. We were able to go from message construct development to integration, test and validation in approximately 2 months with communication between four unique subsystems.

7. AEMU Communication Architecture

Figure 4 shows the AEMU communication architecture block diagram. There are four major subsystems being developed under PAS 2.0. They are the caution warning and control system, informatics, communication, and audio. Also included is a monitor, testing validation system. For PAS 2.0, the video system and the storage system will not be developed. These systems are being considered for future implementation. Each of the systems will be described in detail in the following sections.

The bus structure has been selected to be able to handle the large amounts of data generated from the video subsystem. All the systems are connected via a gigabit ethernet bus (GigE). This ensures sufficient available bandwidth across the bus for all the systems, but specifically to handle multiple streams of high definition video. Each subsystem can be addressed by either IPV4 (Internet protocol version 4) or IPv6. Figure 4 shows the basic addressing scheme selected for both the GigE and wide area networking (WAN) i..e., the radio network.

7.1. Command, Warning and Control

Description

The Caution, Warning, and Control (CWC) includes a major set of subassemblies in the Life Support System (LSS) Control

The CWC Assembly monitors the data channels coming to it from the sensors within the Portable Life Support System (PLSS), and detects out-of-range, or anomalous readings. It then reports these caution and warning items to the crewmember through tones through the same speakers in the helmet as those associated with the Audio Processing subsystem i.e., there is only one set of speakers in PAS-2.0 and all caution and warning signals run through the audio process subassembly.

The CWC Assembly is assigned a Criticality-1S designation, since it is monitoring a life-critical hardware item and is critical for warning the crew of depleted life support resources or safety hazards due to malfunction of the life support system. Any software on the CWC Assembly would be Class-A as defined in NPR 7150.2A [8]

7.1.1. Power Management and Distribution

The Power Management and Distribution (PMAD) subsystem distributes power to the avionics, PLSS, and tools of the EVA system. It includes the battery and recharging interfaces, as well as battery health monitoring. It is currently envisioned that most powered devices will derive power from a single suit power source, as opposed to separate batteries for separate devices.

Since the PMAD Assembly provides power to safety critical systems, the hardware and avionics are considered Criticality-1. Any software in the PMAD Assembly would be Class-A.

The CWC subsystem monitors and controls the Power Management and Distribution subsystem, for discussion purposes - particularly related to software and control - the PMAD, for control purposes is consider part of the CWC - see figure 6.

Implementation

The CWCS is the focus primary development effort regarding form, fit and function at a pre-Preliminary Design Review (PDR) level of maturity for PAS 2.0 [9]. The CWCS is designed to fit in a box approximately 10.25 inches (in.) long x 6.25 in. deep x 3.8 in. high 5.

When fully implemented, the CWCS will provide the following functions:

- Monitor the status of PLSS life-support functions (e.g. consumables and component status) for display to the EVA crewmember
- Provide PLSS fault detection to alert the crewmember of off-nominal conditions and perform corrective actions
- Electronically control the following PLSS components:
- The Suit Water Membrane Evaporator (SWME), which provides heat exchange for the thermal control loop by vaporizing water (H2O) across a water membrane wall
- A fan, which circulates gases through the oxygen ventilation loop
- The Temperature Control Valve (TCV), which is a diverter valve that adjusts the amount of cooling flow to the crewmember, either manually by the crewmember or automatically via a control algorithm
- The Primary Oxygen Regulator (POR), which provides suit pressure regulation at multiple pressure set points of oxygen (O2) delivered from the primary O2 system
- The Secondary Oxygen Regulator (SOR), which provides suit pressure regulation at a single pressure set point of O2 delivered from the secondary O2 system
- The Rapid Cycle Amine (RCA) swing-bed, which provides carbon dioxide (CO2) scrubbing and humidity removal of the ventilation loop

Figure 5: PAS CWCS Packaging Illustration

- A pump, which circulates H2O through the liquid cooling loop to provide crewmember cooling

Operations

In an operational setting, the CWC, it will not receive commands from the outside or from any subsystems. It only sends Events (see Message Types Section) or telemetry (i.e. health and status information). For the development system, one may wish to send messages to the CWC to configure the type of telemetry being sent and the frequency in which various telemetry messages are sent.

One critical item that belongs to CWC is the Caution and Warning Acknowledgement switch (ACK). ACK is used to control caution and warning tones. CWC sends messages to AUDIO indicated warning or emergency and whether or not these have been acknowledged. A lookup table or some other mechanism will be used to determine how long and acknowledgement is valid. It may be that for emergency warnings, one may wish to clear the acknowledgement after some period and force the crew member to re-acknowledge cautions and particularly warnings.

CWC publishes Caution and Warning Report messages. These report message include a time stamp, Caution/Warning type (CW), whether it is a caution or warning, and if this CW has been acknowledged - see Section 8.1, Messages.

Research

The CWC is an engineering development prototype effort. Information obtained from this development effort will be documented for use in developing requirements for future AEMUs. In addition, the AEMU contractors may decide to utilize various internal architectures, and techniques demonstrated here.

7.2. Audio Processing

Description

The Audio Subassembly provides for the handling and control of the audio signals retrieved from the integrated audio processing subassembly located in the helmet and delivers the audio to the communications subsystem or informatics. Likewise, the Audio Subsystem processes audio received from the communications subsystem and delivers that to the integrated audio processing subassembly. The audio subsystem also is required to mix up to four separate inbound audio streams for the communication subsystem along with caution and warning tones and annotations and deliver that to the integrated audio processor D/A speaker subassembly.

The Audio Subsystem interfaces to the Communications subsystem through a GigE bus that is separate from the subsystem GigE bus and delivers microphone array signals prepared to be processed by the Audio Subsystems. Audio data is also sent from the Comm Assembly to this assembly to be delivered to the speakers.

Implementation

Initial implementation was accomplished using an embedded processor development board using an Avnet Spartan-6/OMAP? Co-Processing Development Kit for the outbound voice channel (voice off-suit) and a netbook computer to

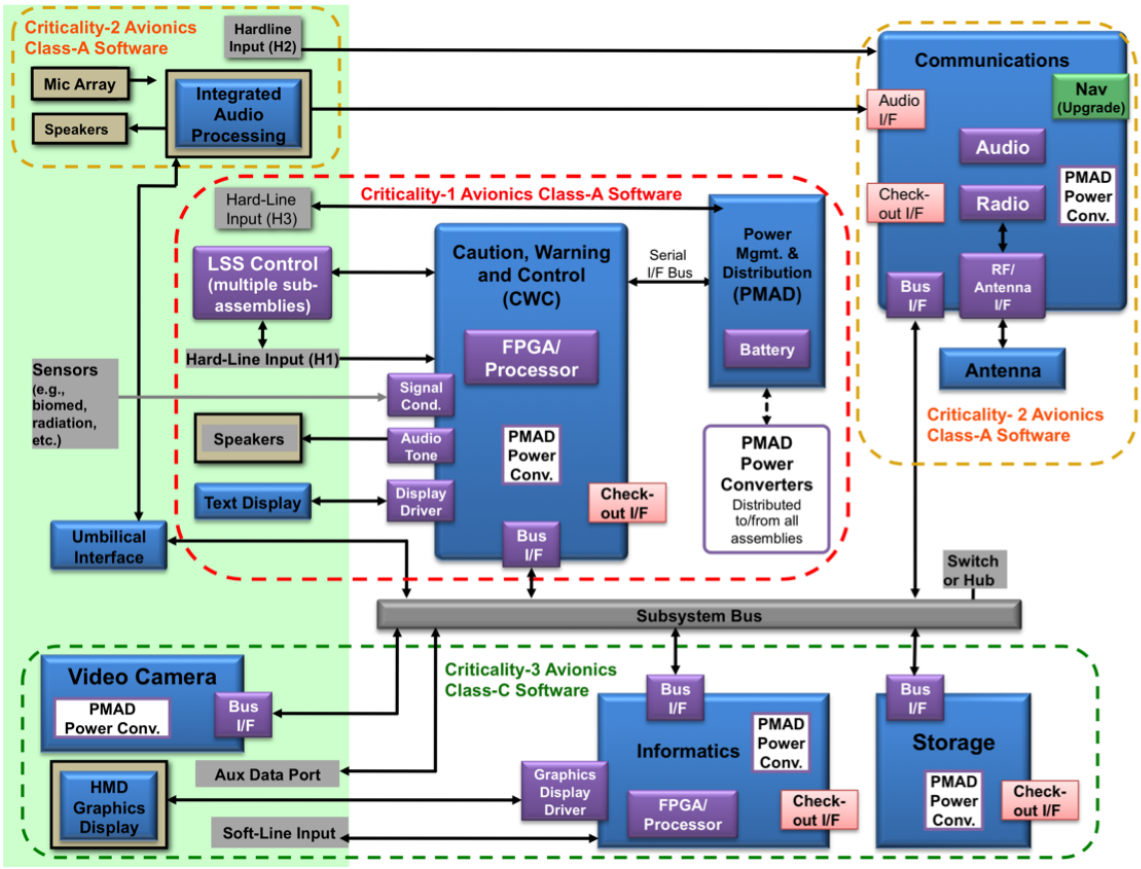

Figure 6: Power, Avionics and Control Software Critical Systems

process the four inbound channels. The GStreamer open source software was used to encode and decode audio streams [10]. This was done simply to have a working audio system available for inter-subsystem message communication testing to test message passing between subsystems as there was approximately 2 months available from conception to integration testing. Four additional netbooks were used to emulate four external audio sources thereby providing four inbound channels. The operating systems on both the embedded processor board and the netbooks was Linux Ubuntu. This allowed all software to be common between the embedded processor and the netbooks. Note, this is not the configuration for audio processing that would be deployed. It was done simply to enable testing of messaging (i.e. telemetry and events). Details of this implementation testing is available in the PAS Subsystem Interface Test Report [11].

Operations

There are three switches associated with the audio system: Mode, Push-to-Talk, and Volume. The is additional switch owned by CWC that directly affects the audio system processing, the Caution and Warning Acknowledgement switch (ACK).

Mode (MODE) - The MODE switch is set for either Push-to-Talk, VOX (voice operated switch) or Open Microphone (Open-Mic).

Push-to-Talk (PTT) - If the MODE is set to PTT and the PTT switch is NOT set, no voice data will flow off suit. If the MODE is set to PTT and the PTT switch IS set, voice data should flow to the appropriate call group(s) as decimated by the communication subsystem Call Group rotary switch.

Volume (VOLUME) - The Volume Switch increases of decreased the overall volume of the combined audio heard

by the crewmenber. This is implemented as an increase slowly while in the UP or increase position and decrease slowly while in DOWN or decrease position. In PAS-2.0, individual in-bound channel volume is controlled via commands from Informatics.

Caution and Warning Acknowledgement (ACK) - ACK is used to control caution and warning tones. CWC sends messages to AUDIO indicated warning or emergency and whether or not these have been acknowledged. This switch will silence those tones. It the caution or warning tones have not been acknowledge, AUDIO will play the tones according to information in a lookup table. That information will include: enunciation, tone frequency, tone repletion period - with zero meaning continuous.

Audio modes include PTT and VOX, and OPEN MIC. You can effectively achieve MUTE by selecting PTT mode on the mode switch and not pushing the PTT button. OPEN MIC is the default. If MODE is set to VOX, only audio that contains voice will flow off suit. In Open-Mic is selected, all audio is sent off suit even if nothing is being said (no voice activation). Thus, the receiving system will hear everything, including just breathing and background noise - assuming this is not filtered in some type of audio processing.

Regarding voice generated on-suit: AUDIO will send Informatics audio in Linear pulse-code modulation (LPCM) encoding format. Informatics can store this for archiving purposes or use it for voice recognition commanding as well as for voice notes. Audio will be sent off-suit according to the position of the corresponding audio switches and the call group switch or some type of indicator. A possible call group is ”Local Only” such that no transmission goes off suit. For safety reasons, this may not be desirable or one may want to be able to over-ride this remotely from HOME or Mission Control.

Since all audio is being place on the internal bus for the Informatics and Communication subsystem to pick up, both the MODE and PTT switches could belong to the Communication subsystem rather than Audio. This may actually simplify processing as no special telemetry messages indicating switch positions have need to be sent from AUDIO to COMM. However, the current approach is to keep these with AUDIO because giving the radio assembly a set of switches related to voice blurs the line between radio and audio functionality. One of the challenges that has occurred in dialogs with Johnson Space Center personnel is the long-ingrained idea that voice = radio/comm. There appears to be difficulty separating the two concepts and adding voice-related HW interface to the radio assembly would only exacerbate this.

For PAS-2.0, the Audio subsystem will locally store all locally generated audio data for one sorties worth of data (for PASA-2.0 the assumption shall be that one sorties is up to 8 hours of data). In future AEMUs implemented with a separate Storage system, audio storage may occur in the storage subsystem rather than locally.

It is possible that there is currently sufficient processing power within the Audio subsystem to perform voice commanding. However, that function is being delegated to Informatics. Thus, all voice must be transmitted to Informatics regardless of any switch positions. This is necessary in order for Informatics to alway hear and interpret voice commands as well as for taking audio notes. A good example of this is current voice recognition software such as Dragon Dictation®. Dragon Dictation allows a soft switch and an audio command. To turn on voice commands one can use the GUI switch or say ”Wake up”. To turn off voice commands or dictation one can also use the GUI switch or say ”Go to sleep”. Thus, Informatics has to receive ALL audio to perceive the ”Wake Up” or ”Go to Sleep” commands.

7.2.1. Inputs

For PAS-2.0, the Audio subsystem will obtain inbound channel gain settings for up to four channels from Informations (or the Test, Monitoring and Validation system).

In order to set the proper audio gains, the Audio subsystem needs static pressure measurements for the suit (helmet). Those measurements come from CWC. For PAS-2.0 these will be periodically transmitted and repeated at a rate sufficient to maintain steady state gain.

For PAS-2.0, the caution and warning tones will be generated within the Audio subsystem along with associated aural annunciations. Notifications come from CWC via messages sent across the GigE bus. One of four warning tone types (advisory/alert, caution/warning and emergency) will be sent for each warning type. The warning types are:

Information - tones used to direct crew attention to messaging of an information nature (i.e. buddy warnings & text messaging)

Status - tones used to identify failures during suit checkout (i.e. failed leak check, failed automated) and other messaging of a status nature

Alert - tones used to identify successful progress during a suit checkout (i.e. start leak check & successful leak check)

Warning - tones uses to identify serious system issues that a crew member needs to take immediate action on. This would include faults during an EVA.

The following are taken from the NASA GRC EVA Power, Avionics, and Software (PAS) Subsystem Architecture Data Book

- Radiation High
- Oxygen Low
- CO_2 High
- Water Low
- Suit Pressure Out of Limit
- Battery Low
- Vent Fan
- Thermal Pump
- H_20 Heater
- LCG Cooling Valve
- Glove Heater

7.2.2. Output

The Audio subsystem will provide various health and status messages. Included in these messages will be switch settings and gain settings for each inbound channel and the local voice channel. Other health ad status messages that may be useful are current and available storage, power consumption, or other measurable hardware related parameters.

For PAS-2.0, detailed system configuration settings are assumed to be available from a configuration file.

As stated earlier, the Audio subsystem will provide voice messages to Informatics for use in voice recognition and audio notes. In future implementations, each message may be tagged with the following information: Time Stamp,VAD (Voice Activated Detection), Gain, MODE setting and PTT and include a payload consisting of nominal 10 ms of LPCM audio. Theses packet are transmitted as unicast UDP packets from AUDIO to INFO.

Outbound streaming voice data from AUDIO to COMM shall be unicast. COMM shall re-transmit to other external nodes according to route tables and call group settings.

For our initial PAS 2.0 interface tests performed in the August of 2014, audio distribution was performed quite differently as describe in the test report [11]. GStreamer audio streams were sent between external systems (e.g. Home and other AEMUs) using notebooks to emulate external systems. All GStreamer packet were sent Unicast and channel (source) separation was performed using a different port number for each source.

Research

To date, the main research activities have been directed at eliminating the comm cap while still maintaining exceptible speech quality. Research includes speech quality and intelligibility tests, audio processing options, and microphone array and placement options. Speech quality and intelligibility tests, audio processing options, and microphone array and placement options. For example: figure 7 shows microphone array on flexible polyimide circuit board. This is a prototype and the actual microphones would be much smaller.

Research is ongoing in DSP-based solutions for audio processing focusing on temporal/spatial processing including:

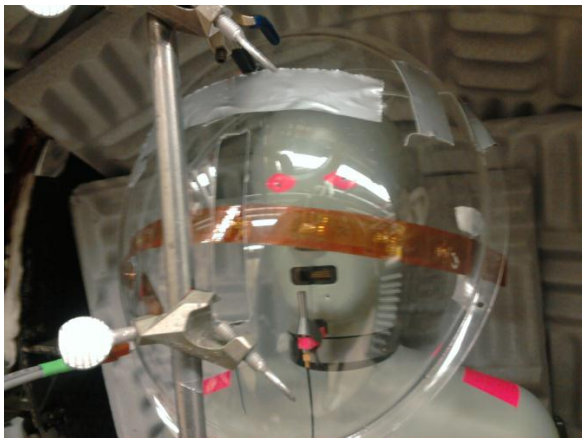

Figure 7: Microphone Array on Flexible Polyimide Circuit Board

- Adaptive beamforming
- Linear beamforming
- Structure borne vibration suppression
- Echo cancellation

Implementation/Processing technologies being investigated and deployed include:

- PGA, DSP processors
- MEMS microphones, Electret microphones
- Analog, digital array interfaces

7.3. Communication

The Communications Subsystem contains the baseband processing and formatting of the digitized data that is transmitted to and received from and to the radio frequency (RF) portion of the communications including digitized voice. The communication subsystem performs the routing of information on and off the AEMU as well as configuration control of the radios.

The Comm Assembly is assigned a Criticality 2R, assuming that if the voice communications are lost during an Extravehicular Activity (EVA), the mission (the EVA) would have to be terminated on that day. If communications could not be repaired, the mission would be seriously jeopardized. Software for the Comm Assembly (and Nav if included as an upgrade) is assumed to be Class-A based on requirements for similar manned systems, and as defined in NPR 7150.2.

7.3.1. Radio

Description

Due to the often conflicting requirements of high data rates and reliable radio coverage, a dual-band radio architecture is proposed for AEMU suit systems. This radio would use a wideband, high data rate Radio Frequency (RF) interface with enough throughput to handle simultaneous voice, telemetry, and video flows, as well as a lower data rate RF interface that can carry mission essential voice and telemetry. The wideband interface would be used amongst all radios while they are in range of each other, and since the range of wideband networks is relatively short, the low-rate interface will be utilized when the wideband interface fails. Non-essential data flows originating from the suit will be queued up locally on the suit and transmitted when the wideband network is once again available. This concept allows

for each of the two interfaces to be designed separately according to each set of requirementsboth coverage and high data rates. [12]

Operations

For PAS-2.0, a dual-band radio was not demonstrated nor any ad-hoc mobile network (MANET) routing. Rather, a single wide-band 802.11 radio broadcast network was utilized (i.e. all radios are within range). The radios were configured for "ad-hoc mode" rather than "infrastructure mode" so that there is no single point of failure (access point failure).

Operationally, there is nothing to do relative to the radios except turn them on or off. There will be health and status data that will be place into telemetry messages. Some meaningful parameters include:

One potential operational consideration is to have the wide-band radio shut down when battery power fall below some threshold. This could be accomplished either via a specific command from the CWC or by monitoring the CWC telemetry power status and taking appropriate action. The later allows reduced software complexity in both the CWC and COMM system while allowing both units to evolve independently.

Research

In the Fall of 2011, the National Aeronautics and Space Administration (NASA) Glenn Research Center (GRC) participated in the Desert Research and Technology Studies (DRATS) field experiments held near Flagstaff, Arizona. A Dual-Band Radio Communication System which was demonstrated during this outing. The EVA radio system was designed to transport both voice and telemetry data through a mobile ad hoc wireless network and employed a dual-band radio configuration [12]. Some key characteristics of this system include:

- Dual-band radio configuration. One radio is used for high data rate communication and the other radio is used for contingencies to provide improved coverage and extended operational range.
- Intelligent switching between two different capability wireless networks. The switching algorithm utilizes the expected transmission count for the wireless links in order to select the appropriate wireless network.
- Self-healing network. Alternative data paths are determined either within the same network, or by transitioning to an alternative independent wireless network with different capabilities and characteristics.
- Simultaneous data and voice communication. The wireless communication system uses Media Access Control (MAC) layer traffic control and open source Speex-based Voice over Internet Protocol (VoIP) technology.

The significant research challenge lies in the implementation of this wireless interface on a hardware platform certified for operation beyond the LEO environment. Currently, a flight-grade chipset that implements a commercial WLAN interface does not exist, and the questions that need to be answered regarding this are what the costs will be for the design and fabrication of an ASIC chipset and also exactly what wireless interface standard will be most suitable for a wide variety of spacecraft proximity communications applications.

Research is ongoing into development of a new lower data rate RF radio waveform to replace the current Space-to-Space EMU Radio (SSER) [13]. The current SSER waveform cannot support new technologies like packetized IP (Internet Protocol) data and its telemetry format is fixed. The current SSER has heritage to the 1970's and is design for ease in integration with the 1553 Bus onboard the spacecraft. Research involves modernizing this waveform to ease with integration of a routed packet network. Some of the issues that need to be addressed are the design of a MAC protocol that maintains the reliability of the SSCS and also meets the requirements for ad-hoc routing, supporting packetized IP, and increasing the robustness of the physical layer by considering things like short/long-term channel estimation feedback, adaptive modulation and coding, and dynamic spectrum access, which are challenging problems in a decentralized radio network. Another desirable feature of this interface is the ability to re-configure the waveform on orbit, so there are also open research issues regarding how to build a flight grade software-defined radio with a low enough SWaP footprint that it may be used on the suit. This will require hardware components with capabilities beyond those used in the software-defined radios developed for the Mars program.

7.3.2. Routing

Description For the general AEMU development, routing should be able to accommodate the generic network scenario shown in figure 4 and described in Desert Research and Technology Studies (DRATS) field experiments report [12]. In general, one should be able to accommodate a multi-homed, dual-band radio with policy-base routing such that only critical voice and telemetry flows over the UHF reliable radio network whereas high-rate science data and video flow over the high-rate radio network. Furthermore, data the radios should allow for multi-hop routing (routing through multiple suits). Much of this has already been demonstrated and new technologies and techniques are being produced at a rapid pace [14] [15].

Operations For PAS-2.0, the IP addressing scheme is provided in figure 4. IPv6 addresses should be used wherever possible as all new commercial Internet technology is expected to utilize IPv6 addressing. This is particularly true for mobile ad hoc networks.

There are currently no plans to implement Delay Tolerant Networking (DTN) on the AEMU and there is currently no standard, and no concept of operations indicates a need within the AEMU. DTN form HOME to wherever could be considered, but there is no demonstrable need for multi-hop store, carry and forward regarding any AEMU scenarios - particularly if no terrestrial EVAs are expected in the foreseeable future.

All IP packets are forwarded on-suit of off-suit according to the forwarding (route) tables. Application that utilize IP addressing should operate without any issues assuming sufficient bandwidth.

There will be a handful of pre-defined Call Groups (a.k.a. voice loops) that the crew members can switch between during a sortie. Comm will translate these loops into IP addresses and transmit the packets accordingly. Those voice loops will be defined by mission control and can be modified at any time. Currently, the plan is to have Call Groups defined in a configuration file. Where the file resided is To-Be-Determined (TBD) and could be HOME. For PAS-2.0, that file will be stored on the Communication system, and could also reside on Informatics. However, since Informatics may not be implemented in the first flight AEMU build, it is better to store that file locally within the routing portion of the communication system.

AUDIO sends its outbound streaming voice data to COMM via unicast IP packets. COMM shall re-transmit to other external nodes according to route tables and call group settings.

Caution and Warning Report messages from CWC shall be published (transmitted) off-suit to the CWC caution and warning subscribers (which is likely all parties). These message should come directly from CWC. In PAS-2.0 they may come form INFO. The report message will include a time stamp, Caution/Warning (CW) type, whether it is a caution or warning, and if this CW has been acknowledged.

Potential Call Groups:

- Local ONLY - No voice leaves the suit. This is for voice commands and audio message stamping of experimental data.
- Broadcast - All parties involved including other AEMU's HOME, and mission control
- Proximity Crew Only (voice between AEMU crew only...this may not be allowed, BTW)
- Mission Ops Only (voice between the crew member and mission ops, which excludes other AEMU crew)

Research

Technology related to multi-home radios and routing likely to move ahead within the commercial arena at a far greater pace and with far greater investment than NASA can provide. Such technology already exists in smartphones and is likely to become standard in vehicle-to-vehicle and vehicle-to-infrastructure communication. Thus, no new research is currently planned related to multi-home, dual-band radio wireless and network interface standards.

7.3.3. Time Syncronization

In PAS-2.0, the communication subsystem is responsible for synchronizing the other subsystems. The current plan is to simply implement Network Time Protocol (NTP) on all subsystems. As timing requirements do not call for high precision, NTP should be sufficient. Furthermore, if NTP is too processor intensive, the time-stamp available for telemetry packets could be used to synchronize subsystems.

During the PAS Interface Testing of August 2013, initial NTP testing showed NTP worked, but that for synchronization of subsystems, it may be better so simply use the time-stamp embedded in all our telemetry packets

- particularly the one from COMM. NTP operates under the pretense that higher ?tiered clock data is available to the lower-tiered clocks for long periods of time [16]. Over time, the NTP slave devices slowly begin to ?trust? the more stable master clocks based on their observed performance, and the slave devices will eventually begin to more aggressively synchronize with these clocks. Since the suit electronics are only powered up for several hours at a time, this process is too long and involved for this application. The behavior of NTP can be and was modified to suit the needs of PAS, but most of these modifications simply disabled a lot of the advanced functionality of NTP. Periodic, forced synchronizations with the COMM timestamp should be more than sufficient to maintain clock synchronization between the PAS assemblies over the duration of an EVA, and the same is true between COMM and its higher-tier clock (AP, Mission Control, etc.).

7.3.4. Navigation

Navigation research related to AEMU development and EVA's is mainly concerned with terrestrial navigation for both safety and to be able to mark locations during science EVAs (i.e. where and when was a particular sample taken).

In preparation for the Constellation Program, NASA was interested in finding new methods of surface navigation to allow astronauts to navigate on the lunar surface. The interest is still there, but in more general terms related to any terrestrial navigation.

In support of the Vision for Space Exploration, the NASA Glenn Research Center developed the Lunar Extra-Vehicular Activity Crewmember Location Determination System and performed testing at the Desert Research and Technology Studies event in 2009. A significant amount of sensor data was recorded during nine tests performed with six test subjects.[17]

Technology related to terrestrial navigation is likely to move ahead within the commercial and military arena at a far greater pace and with far greater investment than NASA can provide. Thus, no new research is currently planned as no terrestrial EVAs are expected any time soon.

7.4. Informatics

The Informatics Assembly is a computerized system aiding crewmembers in their exploration missions. The operational goal is to increase EVA crewmember productivity and effectiveness, and decrease the amount of mission support provided by ground-based controllers.

Functionally, the Informatics Assembly provides information to be a displayed for real-time navigation and tracking information, productivity, and work enhancement aids, such as task procedures, checklists, and schedule coordination. It also provides situational awareness information, such as metabolic rate and consumable usage.

Due to the numerous task Informatics performs, it is expected to generate vast majority of the commands that transition the GigE Bus (e.g., commanding of the Audio subsystem for voice notes, commanding of the video camera for video recording.

The Informatics subsystem performs only non-critical functions and therefore is assigned a Criticality-3 designation and associated software assumed to be Class-C. . Failures within these will not result in a loss of life nor loss of mission. If these systems were down, the crew could continue to perform an EVA, but might not be as productive.

7.4.1. Implementation

For the tabletop breadboard, Informatics was implemented using a Beaglebone development board (http://beagleboard.org/bone). Some of the salient characteristics are:

- TI OMAP processor
- Linux OS
- 720 MHz ARM 7 architecture
- 256 MB RAM
- PowerVR 3D GPU
- File system on micro-SD card (4 GB currently)

7.5. Storage

A separate storage unit is not part of the Power, Avionic and Software (PAS) 2.0 implementation. In PAS 2.0 all subsystems will be responsible for providing their own storage.

The Storage Assembly, is meant to be permanent mass storage for the entire PAS Subsystem, and is therefore non-volatile. This assembly is considered a secondary storage unit and not directly accessible by each of the assembly's processors; rather, each assembly will access it through their input/output channels via the subsystem bus. The Storage Assembly is considered to be all of the associated components that retain the digital data for use by the user and data stored by the PAS Subsystem for archiving.

The storage subsystem is assigned a Criticality-3 designation and associated software is assumed to be Class-C. . Failures within these will not result in a loss of life nor loss of mission. If these systems were down, the crew could continue to perform an EVA, but might not be as productive.

7.6. Video

The Video Camera is a self-contained unit. It is interfaced independently and directly to the PAS Subsystem data bus; therefore, the video data travels on the same bus as other the other PAS data. Thus, this interface requires physical, link, network, and transport layer Open Systems Interconnect (OSI) bus functions. Video imaging is the suits highest data rate source, dwarfing most other suit data by factor of 10 or more.

The lens and sensor for the camera is located on either side of helmet and has a fixed field of view (FOV).

Storage for video is expected to take place in the Storage system for implementations and by Informatics for PAS 2.0 implementation.

The Video Camera Assembly is assumed to be a low criticality device

For the PAS Interface Testing during August of 2013 [11], a commercial-off-the-shelf high definition webcam, a SANYO® VCC-HD4600, was used simply to load the GigE bus as much as possible. No storage was performed.

7.7. Monitoring, Test and Validation

The Monitor, Test and Validation (MTV) system primarily monitors the Gigabit Ethernet (GigE) subsystem bus and the wireless links as well as creating the necessary signals to exercises various subsystems (e.g., subsystem configurations, multiple VOIP streams)

The Validation system displays telemetry in a manner that can be used for debugging bus activity as well as validate test commands.

The Test portion is a packet generation tool to exercise various subsystem commands over the GigE bus. This will be particularly useful for debugging one subsystem independent of another.

For the PAS Interface Testing during August of 2013 [11], the MTV was actually distributed over a number of systems. Informatics has implemented a number of event messages that would normally be sourced by other subsystems in order to test the Informatics display capabilities. Likewise AUDIO implemented four input streams using four netbooks. A separate netbooks was used to monitor the Ethernet bus using Wireshark® and for capturing and playing video from the HD Camera.

8. Intersystem Communications

Design Philosophy

Subsystem communications is performed via a Publish/Subscribe mechanism using ZeroMQ. Request/Respond is discourage as it requires maintaining state and does not appear to provide any advantage over Publish/Subscribe as the Request/Response daemon (program running as a background process) has to be active similar to a Publish/Subscribe daemon. Rather than COMMAND/RESPOND we user PUBLISH/SUBSCRIBE. Where we Publish EVENTS and TELEMETRY. The difference between COMMAND and EVENT is depicted in the following definitions:

EVENT A message sent to inform other subsystems that something happened or to tell other subsystems that you want something done. It is an unsolicited message that does not require an acknowledgement or response. It can be ignored by everyone without consequences. Using an EVENT, a publisher does not know if anyone is listening.

Table 1: Required Fields

sender	integer	A number identifying the sender (1 for CWCS, 4 for informatics, etc)
timestamp	double	The time the message was sent (Unix time plus fractional part, seconds since January 1, 1970)
type	integer	A number identifying the message type (which determines the additional fields in the message)

COMMAND A message that requires an action and an acknowledgement by the receiving party.

Note throughout this document that the term "command" is intentionally avoided as "command" implies request/respond which requires systems to maintain state.

We are using only ZeroMQ publish and subscribe sockets. Messages are defined as message types using Google Protocol Buffers (GPB or simply protocol buffers) in a common aemu.proto file. Each message type has specific parameters associated with it. Those parameters are also defined in the aemu.proto file.

The following are some important notes on ZeroMQ publish/subscribe messages (for a full explanation see: http://zguide.zeromq.org/page:all).:

> In theory with MQ sockets, it does not matter which end connects and which end binds. However, in practice there are undocumented differences that I'll come to later. For now, bind the PUB and connect the SUB, unless your network design makes that impossible.
>
> There is one more important thing to know about PUB-SUB sockets: you do not know precisely when a subscriber starts to get messages. Even if you start a subscriber, wait a while, and then start the publisher, the subscriber will always miss the first messages that the publisher sends. This is because as the subscriber connects to the publisher (something that takes a small but non-zero time), the publisher may already be sending messages out.
>
> This "slow joiner" symptom hits enough people often enough that we're going to explain it in detail. Remember that MQ does asynchronous I/O, i.e., in the background. Say you have two nodes doing this, in this order:
>
> Subscriber connects to an endpoint and receives and counts messages. Publisher binds to an endpoint and immediately sends 1,000 messages.
>
> Then the subscriber will most likely not receive anything. You'll blink, check that you set a correct filter and try again, and the subscriber will still not receive anything.

Note, when using TCP sockets, a publisher does not put anything on the wire until someone subscribes. If you use multicast (pgm or epgm) the publisher will immediately put messages on the wire for anyone to receive.

8.1. Messages

Each message has the structure shown in Figure 3. The required fields are shown in Table 1. These required fields are followed by various parameters, some which are required and some which are optional depending on the particular message. For each message, all the field descriptors identifies are expected to be present. This eases code development as there is no need to search for presence of a particular item. However, this results in additional message types (currently a small number).

The following are the message types taken from the aemu.proto file - see Appendix B. They are split into Implemented Messages and Future Work / Future Considerations. The latter are place holders for functionality the we currently believe requires a message for some controlling source and are thus, generally EVENT type messages. Often, the need (or not) for these messages depends on which subsystem owns and controls buttons and switches. If particular control buttons and switches belong to the subsystem, there is no need for configuration messages to be transmitted between subsystems. However telemetry "Status" messages could still be used by INFO or CWCS to validate switch settings.

8.1.1. Implemented Messages

The following messages were implemented and tested during the Intersystem Communication Testing during August of 2013.
The details of and code for the message, EVENT_BUDDY_INFO message is provide in Appendix A

General Messages
EVENT_TEXT_MESSAGE
EVENT_BUDDY_INFO

CWCS Messages
TELEMETRY_ CWCS_STATUS
TELEMETRY_CWCS_CONSUMABLE_PO2
TELEMETRY_CWCS_CONSUMABLE_SO2
TELEMETRY_CWCS_CONSUMABLE_BATT
TELEMETRY_CWCS_CONSUMABLE_H2O
TELEMETRY_CWCS_CAUTIONWARNING
TELEMETRY_CWCS_PHYSIO
TELEMETRY_CWCS_BASICS
TELEMETRY_CWCS_TWO_LINE_DISPLAY

COMM Messages
TELEMETRY_COMM_STATUS

AUDIO Messages
TELEMETRY_AUDIO_STATUS
TELEMETRY_AUDIO_STATUS_OUTBOUND
TELEMETRY_AUDIO_STATUS_INBOUND
EVENT_PLAY_AUDIO
EVENT_AUDIO_SET_VOLUME_CHANNELS
EVENT_AUDIO_SET_VOLUME_TONES

INFO Messages
TELEMETRY_INFO_STATUS

8.1.2. Future Work / Desired Functionality

These are functionality we believe need to exist, but are unsure who owns the control buttons and switches and, thus, we don't yet know if the message is necessary or who sources the message.

COMM Messages
EVENT_COMM_CONFIG_REBOOT (Deprecated)
EVENT_COMM_CONFIG_UPDATE
EVENT_COMM_CONFIG_CRITICAL_RADIO
EVENT_COMM_CONFIG_HIGH_RATE_RADIO
EVENT_COMM_CONFIG_CALL_GROUP_RADIO

System and subsystem reboot message were considered. However, if a subsystem goes down, it probably cannot receive a reboot message. Rather, one would probably have to reboot via some type of power cycle. Thus, the notion or rebooting subsystem appears to be valid, but how to actually accomplish this has yet to be resolved.

The communication system configurations consists for routing information and radio configuration. The radio configuration could include data rates, modulation formats, media access types (e.g. time-division-multiplexed or

frequency-division-multiplexed) and perhaps a variety of waveforms. Who controls this waveforms of configurations is to-be-determined (TBD). The initial thought is that this is a MOC function.

Call group configuration may be controlled by MOC. Or, MOC may configure the possible call groups with the crewmember controlling which call group they are participating in via a switch on the radio or via voice-commanding.

References

[1] NASA. Console handbook - extravehicular activity systems [online].

[2] Google protocol buffers.
URL https://developers.google.com/protocol-buffers/docs/overview

[3] P. Aimonen, Nanopb - protocol buffers with small code size.
URL http://koti.kapsi.fi/jpa/nanopb/

[4] D. A. Joseph, D. Chap, Auto-generate wireshark/ethereal dissector plugins for protocol buffer messages.
URL http://code.google.com/p/protobuf-wireshark/

[5] The intelligent transport layer [cited 13-July-2013].
URL http://www.zeromq.org/

[6] Gnu lesser general public license [cited 13-July-2013].
URL @article{zmq,Date-Added={2013-07-1114:28:52+0000},Date-Modified={2013-07-1114:31:57+0000},Keywords={software,transport},Lastchecked={13-July-2013},Month={July},Title={TheIntelligentTransportLayer},Url={http://www.zeromq.org/},Year={2013}}http://www.gnu.org/licenses/lgpl.html

[7] A. Dworak, M. Sobczak, F. Ehm, W. Sliwinski, P. Charrue, Middleware trends and market leaders 2011, Tech. rep. (2011).

[8] O. of the Chief Engineer, Nasa software engineering requirements NPR7150.2A.

[9] Caution, warning, and control subsystem (cwcs) pre-delivery functional test plan and procedures, Tech. Rep. PAS-018, NASA Glenn Research Center (April 2013).

[10] gstreamer.freedesktop.org, Gstreamer open source multimedia framework [cited 2013].
URL http://gstreamer.freedesktop.org/

[11] W. Ivancic, O. S. Sands, C. J. Bakula, D. Oldham, T. Wright, M. Bradish, J. Klebau, Power avionics and software interface test report - 2013, Nasa technical memoradum, NASA Glenn Research Center (August 2013).

[12] A. J. Swank, C. J. Bakula, Eva radio drats 2011 report.

[13] National Aeronautics and Space Administration,, Lyndon B. Johnson Space Center, Houston, Texas, Extravehicular Activities Space-To-Space Communication System Training Workbook (JSC-36345) (March 2000).

[14] U. D. of Transportation, Connected vehicle research [cited 2013].
URL http://www.its.dot.gov/connected_vehicle/connected_vehicle.htm

[15] etsi.org, Intelligent transportations system [cited 2013].
URL http://www.etsi.org/technologies-clusters/technologies/intelligent-transport

[16] D. Mills, J. Martin, J. Burbank, W. Kasch, Network Time Protocol Version 4: Protocol and Algorithms Specification, RFC 5905 (Proposed Standard) (Jun. 2010).
URL http://www.ietf.org/rfc/rfc5905.txt

[17] D. Chelmins, O. S. Sands, A. Swank, Data analysis techniques for a lunar surface navigation system testbed.

Appendix A. Example Message

The following is an example of a message, the EVENT_BUDDY_INFO message.

This asynchronous message is intended to be sent off suit and should be subscribed to by all other interested parties - particularly other crew members. The intent is to allow crew members to see each others system status. In an operational system EVENT_BUDDY_INFO would be generated by CWCS. For the Interoperability Communication Testing of August 2013, this message was created by the the Informatics system operating in the capacity of a message generator portion of the Test, Validation and Monitoring subsystem. It is anticipated that this message could be subscribed to from the Informatics subsystems on other AEMUs, HOME where HOME is where the AEMU returns after an EVA, and the Mission Operations Center (MOC).

For PAS-2.0, Informatics is the only subsystem that subscribes to EVENT_BUDDY_INFO. Upon receiving EVENT_BUDDY_INFO message, Informnatics updates its displays and to sends and EVENT_PLAY_AUDIO messages to AUDIO when an EVENT_BUDDY_INFO message is received from off-suite. This allows the AUDIO subsystem to play the proper caution and warning tones. Note, AUDIO does not listen for buddy messages. Rather, AUDIO is sent the EVENT_PLAY_AUDIO messages from it's Informatics assembly to notify the crew member of EVENT_BUDDY_INFO message receipt.

Note, "TimeLeft" is in seconds.

```
aemu_Message msg;
memset(&msg, 0, sizeof(aemu_Message)); /* default all fields to false */
/* required fields */
msg.sender = aemu_Message_Source_INFO;
msg.timestamp = timeNow();
msg.type = aemu_Message_Type_EVENT_BUDDY_INFO;
/* fields needed for EVENT_BUDDY_INFO (with nanopb, .has_XXX = true is needed for active fields tagged as optional) */
msg.has_suitID = true;
msg.suitID = 4; /* int32, identifier to tell assets apart */
msg.has_po2TimeLeft = true;
msg.po2TimeLeft = 350; /* int32, primary oxygen time remaining in seconds */
msg.has_so2TimeLeft = true;
msg.so2TimeLeft = 5000; /* int32, secondary oxygen time remaining in seconds */
msg.has_battTimeLeft = true;
msg.battTimeLeft = 750; /* int32, battery time remaining in seconds */
msg.has_h2oTimeLeft = true;
msg.h2oTimeLeft = 4958; /* int42, water time remaining in seconds */
msg.has_latitude = true;
msg.latitude = 41.41029; /* float, suit position latitude in decimal degrees North */
msg.has_longitude = true;
msg.longitude = -81.86486; /* float, suit position longitude in decimal degrees East */
msg.has_altitude = true;
msg.altitude = 818.0; /* float suit position altitude in feet above sea level */
/* 0 to 7 caution and warning reports to send to buddies */
msg.cwReport_count = 1; /* number of caution and warning reports in this message */
msg.cwReport[0].has_savedTimestamp = true;
msg.cwReport[0].savedTimestamp = 13857834.9; /* double, time stamp of the caution or warning */
msg.cwReport[0].has_warningType = true;
msg.cwReport[0].warningType = aemu_Message_WarningType_PO2_LOW;
msg.cwReport[0].has_isCaution = true;
msg.cwReport[0 ].isCaution = false ; /* bool, true means caution, false means warning */
```

msg.cwReport[0].has_isAcked = true;
msg.cwReport[0].isAcked = false ; /* bool, true means acknowledged, false means not acknowledged */

Appendix B. Google Protocol Buffers aemu.proto file

```
    package aemu;

    message Message {
// required fields are kept to a minimum (3) because they must appear in every message

    enum Source {
GEN = 0; // general messages not associated with a subsystem
CWCS = 1; // Caution and Warning Control
COMM = 2;
AUDIO = 3;
INFO = 4; // Informatics
VIDEO = 5; // placeholder
STORAGE = 6; // placeholder
MTV = 9; // monitor, test, validation
}
required Source sender = 1; // appears in every message

required double timestamp = 2; // appears in every message; the time the message was sent [Unix time plus fractional part, seconds since January 1, 1970]

enum Type
// GEN types range from 0-399 ——————

NOP = 0; // ”No Operation”
EVENT_TEXT_MESSAGE = 1;
EVENT_BUDDY_INFO = 2;
EVENT_LOCATION = 3;
EVENT_DIRECTION = 4;

// CWCS types range from 400-799 ——————

TELEMETRY_CWCS_STATUS = 400;

TELEMETRY_CWCS_CONSUMABLE_PO2 = 401;
TELEMETRY_CWCS_CONSUMABLE_SO2 = 402;
TELEMETRY_CWCS_CONSUMABLE_BATT = 403;
TELEMETRY_CWCS_CONSUMABLE_H2O = 404;

TELEMETRY_CWCS_CAUTIONWARNING = 405;

TELEMETRY_CWCS_PHYSIO = 406;

TELEMETRY_CWCS_BASICS = 407;

TELEMETRY_CWCS_TWO_LINE_DISPLAY = 408;
```

// COMM types range from 800-1199 ————————

TELEMETRY_COMM_STATUS = 800;
EVENT_COMM_CONFIG_REBOOT = 801;
EVENT_COMM_CONFIG_UPDATE = 802;
EVENT_COMM_CONFIG_CRITICAL_RADIO = 803;
EVENT_COMM_CONFIG_HIGH_RATE_RADIO = 804;
EVENT_COMM_CONFIG_CALL_GROUP = 805;

// AUDIO types range from 1200-1599 ————————

TELEMETRY_AUDIO_STATUS = 1200;
TELEMETRY_AUDIO_STATUS_INBOUND = 1201;
TELEMETRY_AUDIO_STATUS_OUTBOUND =1202;
EVENT_PLAY_AUDIO = 1203;
EVENT_AUDIO_SET_VOLUME_CHANNELS = 1204;
EVENT_AUDIO_SET_VOLUME_TONES = 1205;

// INFO types range from 1600-1999 ————————

TELEMETRY_INFO_STATUS = 1600;
}
required Type type = 3; // appears in every message; determines which optional fields should be present

// additional optional fields depend on type ————————————————-

// GEN types range from 10-399

optional string broadcastMessage = 10; // used by type TEXT_MESSAGE_BROADCAST

optional int32 genNodeID = 11; // used by type BUDDY_INFO_BROADCAST and TELEMETRY_COMM_STATUS; unique system ID
optional int32 po2TimeLeft = 12; // used by type BUDDY_INFO_BROADCAST; primary oxygen time left [seconds]
optional int32 so2TimeLeft = 13; // used by type BUDDY_INFO_BROADCAST; secondary oxygen time left [seconds]
optional int32 battTimeLeft = 14; // used by type BUDDY_INFO_BROADCAST; battery time left [seconds]
optional int32 h2oTimeLeft = 15; // used by type BUDDY_INFO_BROADCAST; water time left [seconds]
optional float latitude = 16; // used by type BUDDY_INFO_BROADCAST and EVENT_LOCATION; position [decimal degrees, negative for South]
optional float longitude = 17; // used by type BUDDY_INFO_BROADCAST and EVENT_LOCATION; position [decimal degrees, negative for West]
optional float altitude = 18; // used by type BUDDY_INFO_BROADCAST and EVENT_LOCATION; position [meters above sea level]
message CautionAndWarningReport
optional double savedTimestamp = 1;
optional WarningType warningType = 2;
optional bool isCaution = 3;
optional bool isAcked = 4;

repeated CautionAndWarningReport cwReport = 19; // used by type BUDDY_INFO_BROADCAST

optional float heading = 20; // used by type EVENT_DIRECTION; degrees from north (east is positive)

```
// CWC starts at 400 - 799;

enum Status {
UNKNOWN = 0;
NOMINAL = 1;
// ...
}
optional Status status = 400; // used by type TELEMETRY_CWCS_STATUS and TELEMETRY_COMM_STATUS
and TELEMETRY_AUDIO_STATUS and TELEMETRY_INFO_STATUS

optional int32 timeRemaining = 401; // used by type TELEMETRY_CWCS_CONSUMABLE_PO2 and TELEME-
TRY_CWCS_CONSUMABLE_SO2 and TELEMETRY_CWCS_CONSUMABLE_BATT and TELEMETRY_CWCS_CONSUMABLE
[seconds]
optional float currentPrimary = 402; // used by type TELEMETRY_CWCS_CONSUMABLE_PO2 and TELEME-
TRY_CWCS_CONSUMABLE_SO2 and TELEMETRY_CWCS_CONSUMABLE_BATT and TELEMETRY_CWCS_CONSUMABLE
current amount
optional float maxPrimary = 403; // used by type TELEMETRY_CWCS_CONSUMABLE_PO2 and TELEMETRY_CWCS_CONSUM
and TELEMETRY_CWCS_CONSUMABLE_BATT and TELEMETRY_CWCS_CONSUMABLE_H2O; maximum amount
optional float critPrimary = 404; // used by type TELEMETRY_CWCS_CONSUMABLE_PO2 and TELEMETRY_CWCS_CONSUMA
and TELEMETRY_CWCS_CONSUMABLE_BATT and TELEMETRY_CWCS_CONSUMABLE_H2O; critical value

enum WarningType {
GENERIC = 0;
PO2_LOW = 1;
SO2_LOW = 2;
BATT_LOW = 3;
H2O_LOW = 4;
}
optional WarningType warningType = 405; // used by type TELEMETRY_CWCS_CAUTIONWARNING
optional bool isCaution = 406; // used by type TELEMETRY_CWCS_CAUTIONWARNING
optional bool isAcked = 407; // used by type TELEMETRY_CWCS_CAUTIONWARNING
optional string warningDetails = 408; // used by type TELEMETRY_CWCS_CAUTIONWARNING

optional float metabolicRate = 409; // used by type TELEMETRY_CWCS_PHYSIO; in BTU/hour
optional float heartRate = 410; // used by type TELEMETRY_CWCS_PHYSIO; in beats/minute

enum CwcsMode {
MODE_MANUAL = 0;
MODE_EVA_43 = 2;
MODE_EVA_60 = 3;
MODE_EVA_82 = 4;
}
optional CwcsMode cwcsMode = 412; // used by type TELEMETRY_CWCS_BASICS
optional float evaTime = 411; // used by type TELEMETRY_CWCS_BASICS; time since start of EVA [seconds]
optional float suitPressure = 413; // used by type TELEMETRY_CWCS_BASICS; [psi]
optional float feedwaterPressure = 414; // used by type TELEMETRY_CWCS_BASICS; [psi]
optional float waterTemperature = 415; // used by type TELEMETRY_CWCS_BASICS; [degrees F]
optional float batteryVoltage = 416; // used by type TELEMETRY_CWCS_BASICS; [Volts DC]
optional float batteryCurrent = 417; // used by type TELEMETRY_CWCS_BASICS; [Amps]

optional string twoLineDisplay1 = 418; // used by type TELEMETRY_CWCS_TWO_LINE_DISPLAY
```

optional string twoLineDisplay2 = 419; // used by type TELEMETRY_CWCS_TWO_LINE_DISPLAY

// COMM starts at 800

optional int32 commOperationalStatus = 801; // used by type TELEMETRY_COMM_STATUS; everything within range
optional string commFirmware = 802; // used by type TELEMETRY_COMM_STATUS; Version Info
optional string commSoftware = 803; // used by type TELEMETRY_COMM_STATUS; Version Info
optional string commWaveform = 804; // used by type TELEMETRY_COMM_STATUS; Version Info
optional bool commExtTimeSync = 805; // used by type TELEMETRY_COMM_STATUS; COMM time synced to external source YES = 1, NO = 0
optional int32 commSysHardwareFailure = 806; // used by type TELEMETRY_COMM_STATUS; Zero, on failure. All other values are failure codes
optional bool commCriticalRadioOn = 812; // used by type TELEMETRY_COMM_STATUS; ON =1, OFF = 0
optional bool commCriticalRadioSync =813; // used by type TELEMETRY_COMM_STATUS; 1 = Synced, 0 = Loss of Sync
optional bool commCriticalRadioTracking =814; // used by type TELEMETRY_COMM_STATUS; 1 = Locked, 0 = Loss of Lock
optional int32 commCriticalRadioSignalStrength = 815; // used by type TELEMETRY_COMM_STATUS
optional int32 commCriticalRadioDataRate = 816; // used by type TELEMETRY_COMM_STATUS
optional float commCriticalRadioBerRate = 817; // used by type TELEMETRY_COMM_STATUS
optional float commCriticalRadioSN = 818; // used by type TELEMETRY_COMM_STATUS
optional bool commHighRateRadioOn = 822; // used by type TELEMETRY_COMM_STATUS; ON =1, OFF = 0
optional bool commHighRateRadioSync =823; // used by type TELEMETRY_COMM_STATUS; 1 = Synced, 0 = Loss of Sync
optional bool commHighRateRadioTracking =824; // used by type TELEMETRY_COMM_STATUS; 1 = Locked, 0 = Loss of Lock
optional int32 commHighRateSignalStrength = 825; // used by type TELEMETRY_COMM_STATUS
optional int32 commHighRateDataRate = 826; // used by type TELEMETRY_COMM_STATUS
optional float commHighRateBerRate = 827; // used by type TELEMETRY_COMM_STATUS
optional float commHighRateSN = 828; // used by type TELEMETRY_COMM_STATUS; perhaps should be Eb/No rather than S/N
optional bool commHighRateHardwareFailure = 829; // used by type TELEMETRY_COMM_STATUS; Failure = 1

// Routing Info
enum CallGroup {
LOCAL_ONLY = 0; // No voice leaves suit
BROADCAST = 1; // All parties including mission ops, HOME and other AEMUs
PROXIMITY = 2; // All parties not including mission ops (e.g., HOME and other AEMUs)
MISSION_OPS_ONLY = 3; // Peer session with Mission Ops, no other parties
}
optional CallGroup commCallGroup = 850; // used by type EVENT_COMM_CONFIG_CALL_GROUP

optional string commConfigUpdate = 852; // used by type EVENT_COMM_CONFIG_UPDATE; configuration file to load

optional bool commSetCritalRadioOn = 853; // used by type EVENT_COMM_CONFIG_CRITICAL_RADIO; true enables radio, false disables radio

optional bool commSetHighRateRadioOn = 854; // used by type EVENT_COMM_CONFIG_HIGH_RATE_RADIO; true enables radio, false disables radio

// AUDIO starts at 1200;

optional int32 audioOperationStatus = 1201; // used by type TELEMETRY_AUDIO_STATUS; zero means everything within range, other numbers have other meanings
optional string audioFirmware = 1202; // used by type TELEMETRY_AUDIO_STATUS; Version Info
optional string audioSoftware = 1203; // used by type TELEMETRY_AUDIO_STATUS; Version Info
optional int32 audioHardwareFailure = 1204; // used by type TELEMETRY_AUDIO_STATUS; Zero, on failure, all other values are failure codes

enum AudioMode {
PTT = 0; // PTT switch is operational and take precidence
VOX = 1; // Voice is sent relative to Voice Activation Detection (VAD)
OPEN_MIC = 2; // Voice is sent regardless of VAD (silence or breathing or whatever)
}
optional AudioMode audioMode = 1212; // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND
optional float audioGain = 1210; // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND; pressure compensated microphone gain [dB]
optional bool audioPTT = 1213; // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND; OFF = 0, ON = 1
optional float audioVolume = 1214; // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND; 0 - 10 for gstreamer
optional float audioOVDC = 1215; // used by type TELEMETRY_AUDIO_STATUS_OUTBOUND; output voice duty cycle

optional float audioVolumeChannel0 = 1220; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional bool audioMuteChannel0 = 1221; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeChannel1 = 1222; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional bool audioMuteChannel1 = 1223; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeChannel2 = 1224; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional bool audioMuteChannel2 = 1225; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeChannel3 = 1226; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional bool audioMuteChannel3 = 1227; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeChannel4 = 1228; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional bool audioMuteChannel4 = 1229; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional float audioVolumeTones = 1250; // used by type TELEMETRY_AUDIO_STATUS_INBOUND
optional bool audioMuteTones = 1251; // used by type TELEMETRY_AUDIO_STATUS_INBOUND

optional float audioSetVolumeChannel0 = 1230; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional bool audioSetMuteChannel0 = 1231; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional float audioSetVolumeChannel1 = 1232; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional bool audioSetMuteChannel1 = 1233; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional float audioSetVolumeChannel2 = 1234; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional bool audioSetMuteChannel2 = 1235; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional float audioSetVolumeChannel3 = 1236; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional bool audioSetMuteChannel3 = 1237; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional float audioSetVolumeChannel4 = 1238; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS
optional bool audioSetMuteChannel4 = 1239; // used by type EVENT_AUDIO_SET_VOLUME_CHANNELS

optional float audioSetVolumeTones = 1252; // used by type EVENT_AUDIO_SET_VOLUME_TONES
optional bool audioSetMuteTones = 1253; // used by type EVENT_AUDIO_SET_VOLUME_TONES; probably never used, not allowed.

// if only one sounds can play at a time, higher numbers have higher priority
enum AudioPlaySound {

```
AUDIO_TONE_INFORMATION = 0;
AUDIO_TONE_STATUS = 1;
AUDIO_TONE_ALERT = 2;
AUDIO_TONE_WARNING = 3;
}
optional AudioPlaySound audioPlaySound = 1260; // used by type EVENT_PLAY_AUDIO
optional bool audioPlayStart = 1261; // used by type EVENT_PLAY_AUDIO
}
```